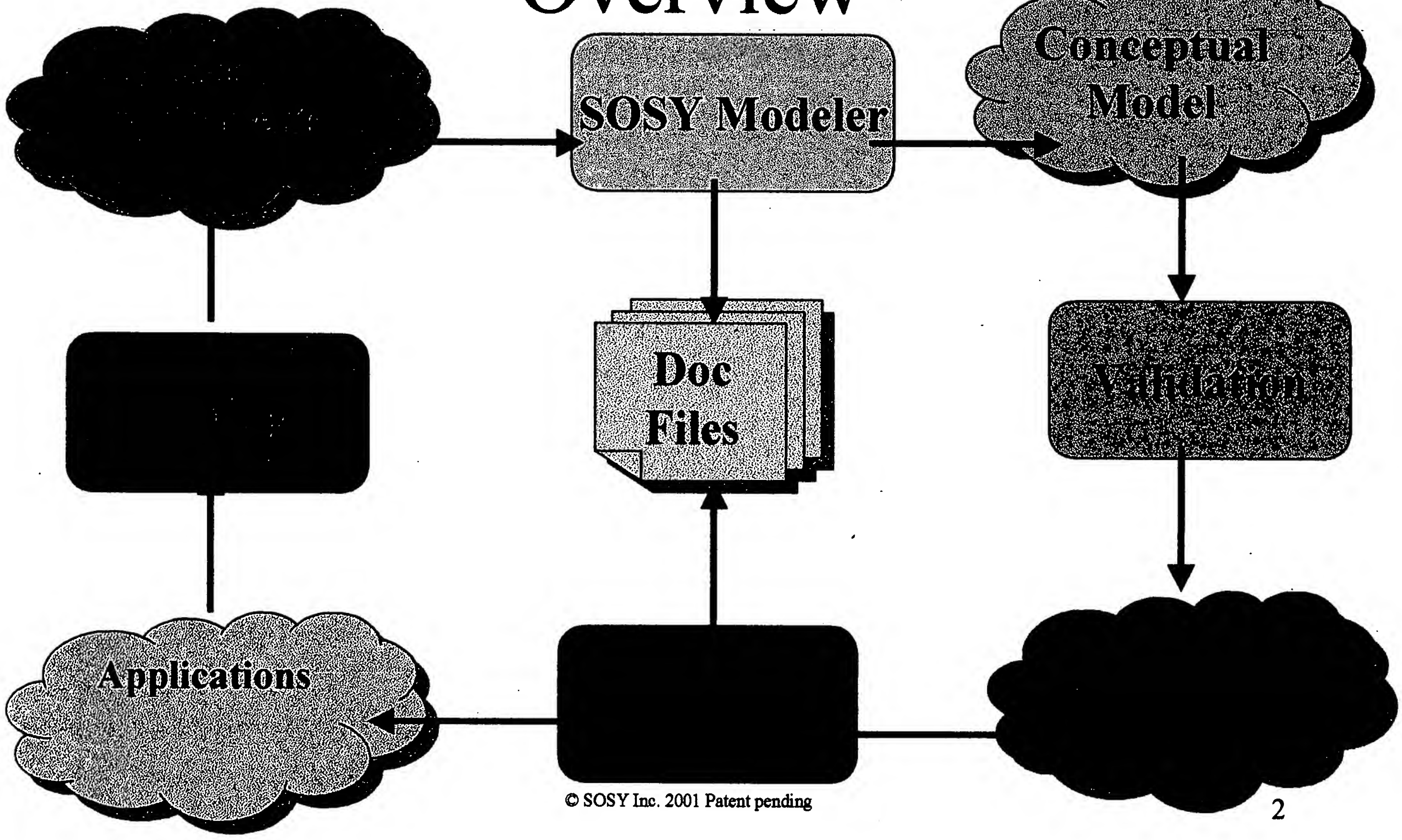
Summary

- Modelling
- Validation
- Documentation

- Persistence
- Bussiness Logic
- User Interface

Overview



© SOSY Inc. 2001 Patent pending

APPENDIX A

Conceptual Modeling Phase

CARE Technologies, S.A.

Index

- Intro
- Overview
- Phase 0. Requirements elicitation.
- Phase 1. Classes identification.
- Phase 2. Relationships between classes.
- Phase 3. Filling classes' details.

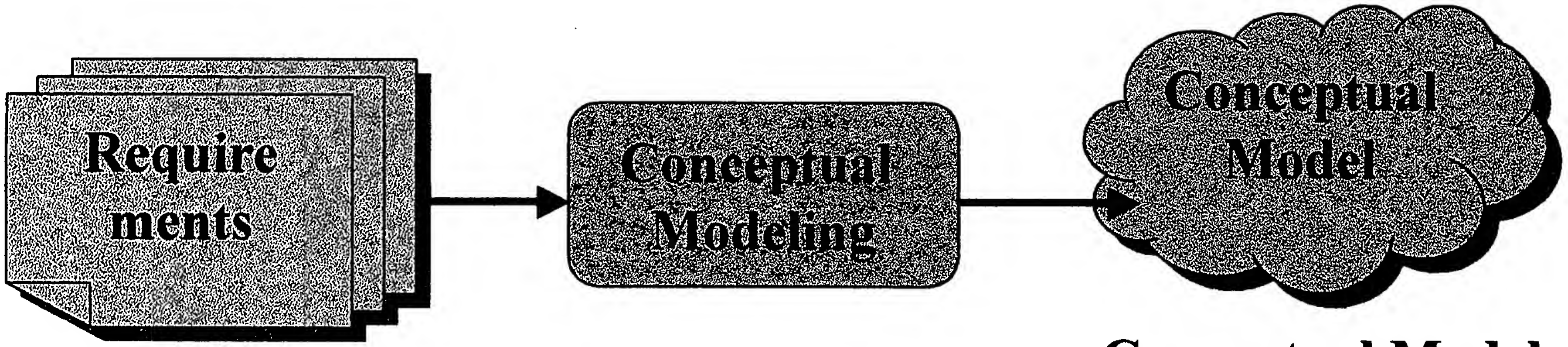
Index

- Phase 4. Express evaluations.
- Phase 5. Agent relationships.
- Phase 6. State Transition Diagram.
- Phase 7. Presentation Model.

Intro

- Conceptual Modeling Phase is a process of systematically & precisely description of the system to build, using:
  - Graphical UML compliant diagrams.
  - Constrains and semantics in a formal non-ambiguous language.
  - This phase is assisted by an integrated Modeler tool.

Overview



APPENDIX A

Requirements

- Specifications
- Documents
- Interviews
- Reports
- Other info. sources

Conceptual Model

- Classes
- Relationships
- Attributes
- Services
- …

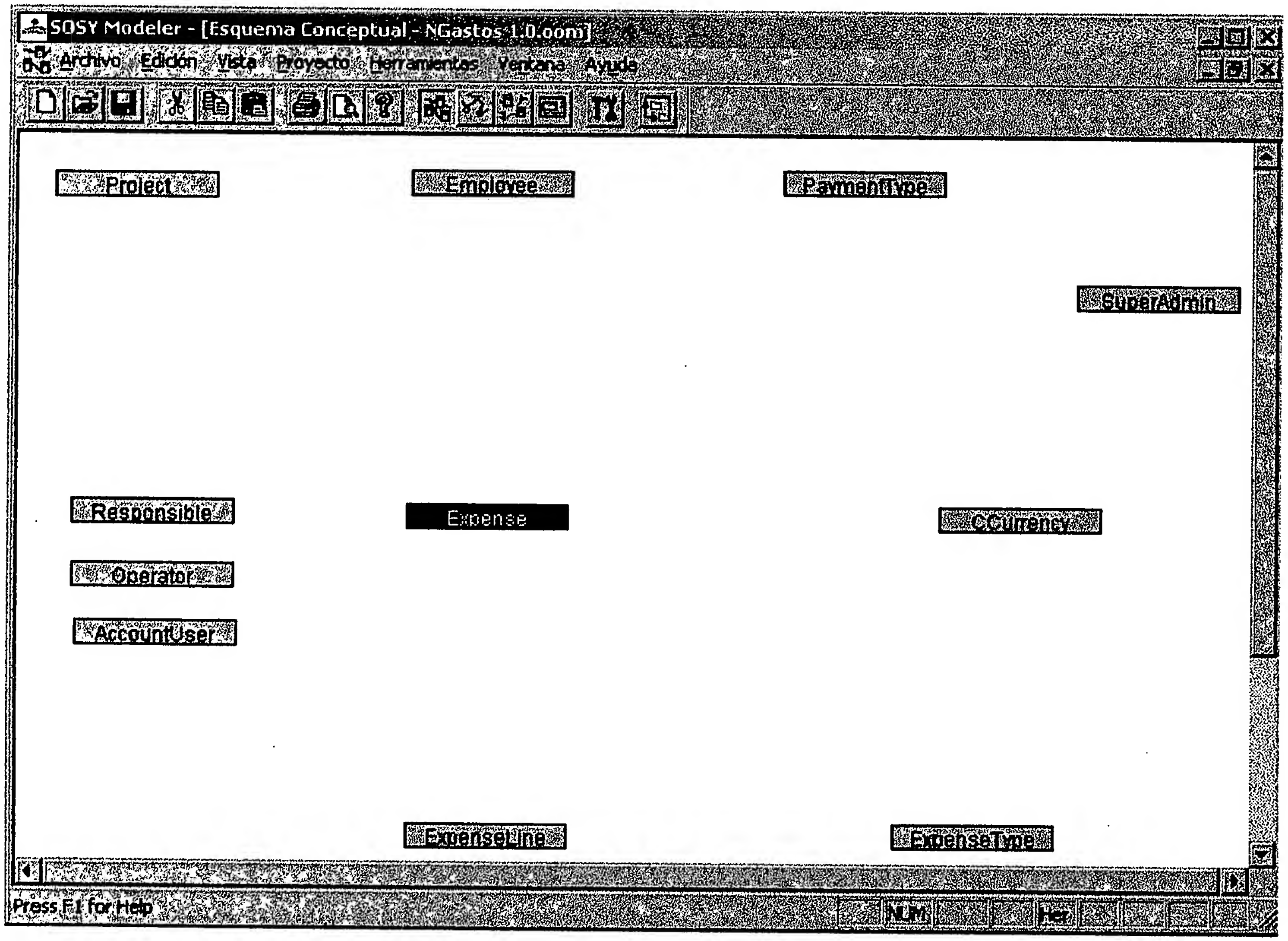
Expressed in a non-ambiguous language.

Phase 0. Requirement elicitation.

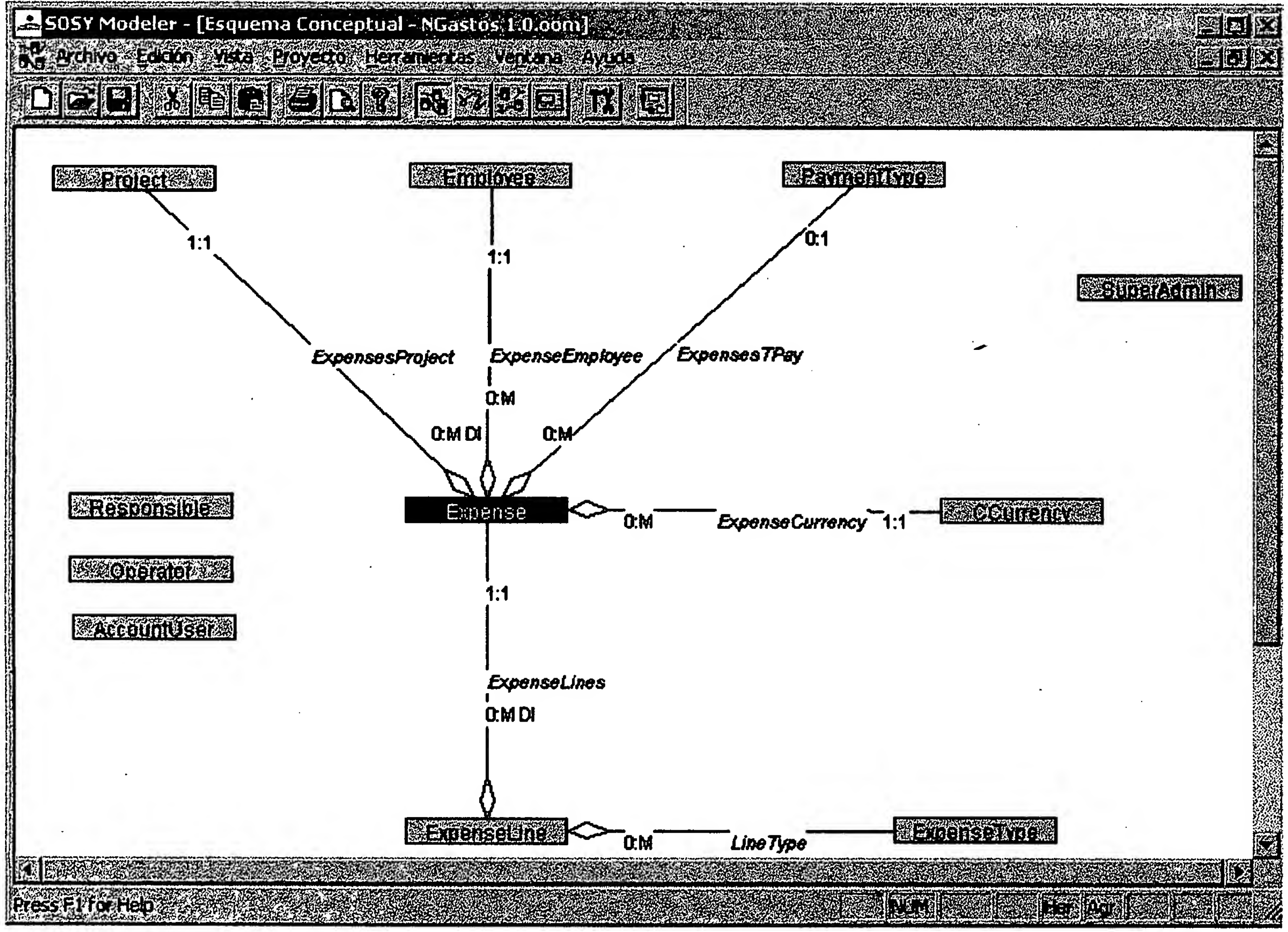
- Gathering the system requirements.
  - By meetings & interviews with customers, experts and final users.
  - By collecting reports, or documents expressing the system how-to and using tools.
  - Obtaining a coherent set of information as input to the next phase.

APPENDIX A

Phase 1. Classes identification.

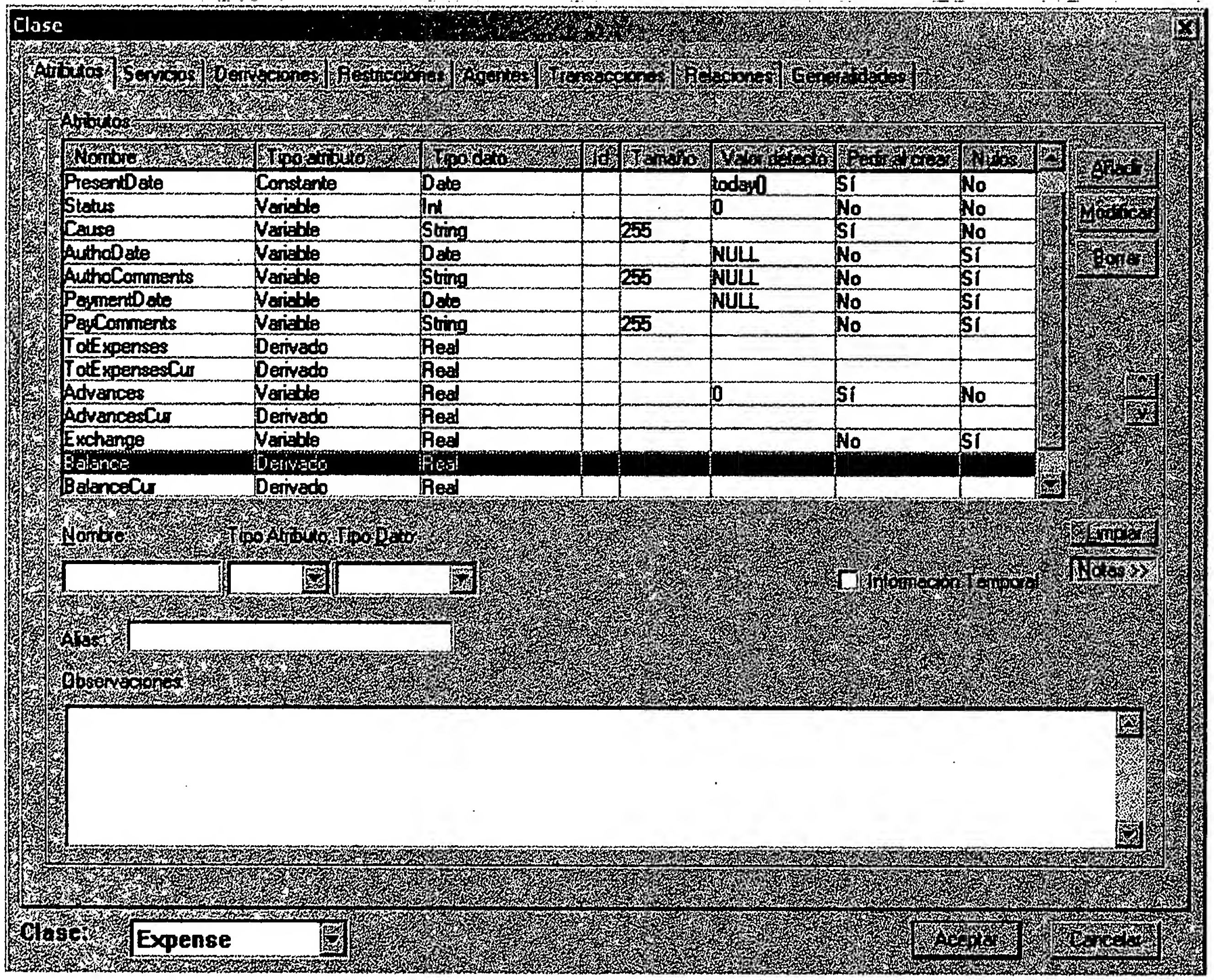


Phase 2. Relationships between classes.



APPENDIX A

Phase 3. Filling classes' details.



Atributos

Nombre	Tipo atributo	Tipo dato	Id	Tamaño	Valor defecto	Pedir al crear	Nulos
PresentDate	Constante	Date			today()	Sí	No
Status	Variable	Int			0	No	No
Cause	Variable	String		255		Sí	No
AuthoDate	Variable	Date			NULL	No	Sí
AuthoComments	Variable	String		255	NULL	No	Sí
PaymentDate	Variable	Date			NULL	No	Sí
PayComments	Variable	String		255		No	Sí
TotExpenses	Derivado	Real					
TotExpensesCur	Derivado	Real					
Advances	Variable	Real			0	Sí	No
AdvancesCur	Derivado	Real					
Exchange	Variable	Real				No	Sí
Balance	Derivado	Real					
BalanceCur	Derivado	Real					

Nombre: Tipo Atributo: Tipo Dato:

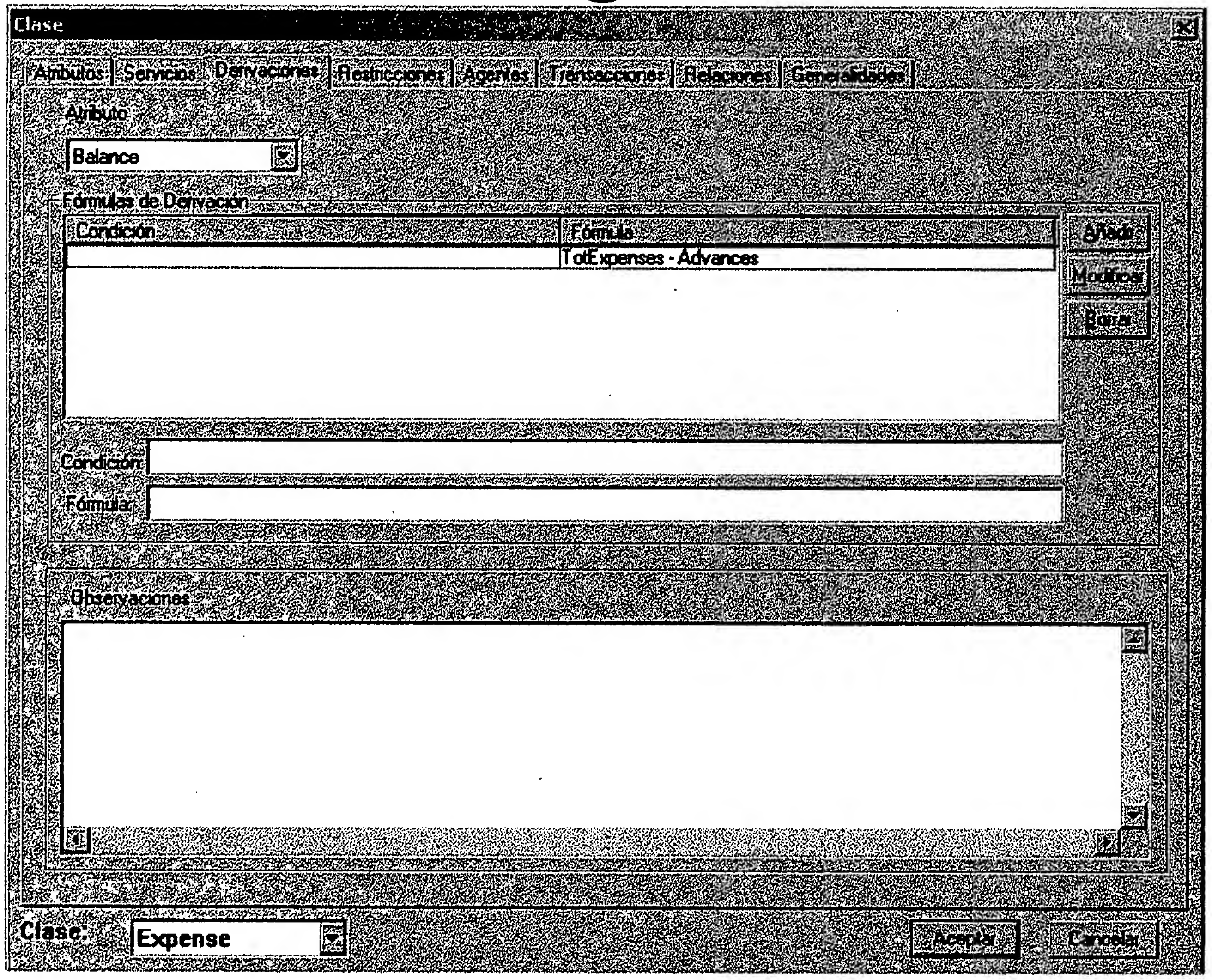
☐ Información Temporal

Alias:

Observaciones:

Clase: Expense

Phase 3. Filling classes' details.



Atributo: Balance

Fórmulas de Derivación

Condición	Fórmula
	TotExpenses - Advances

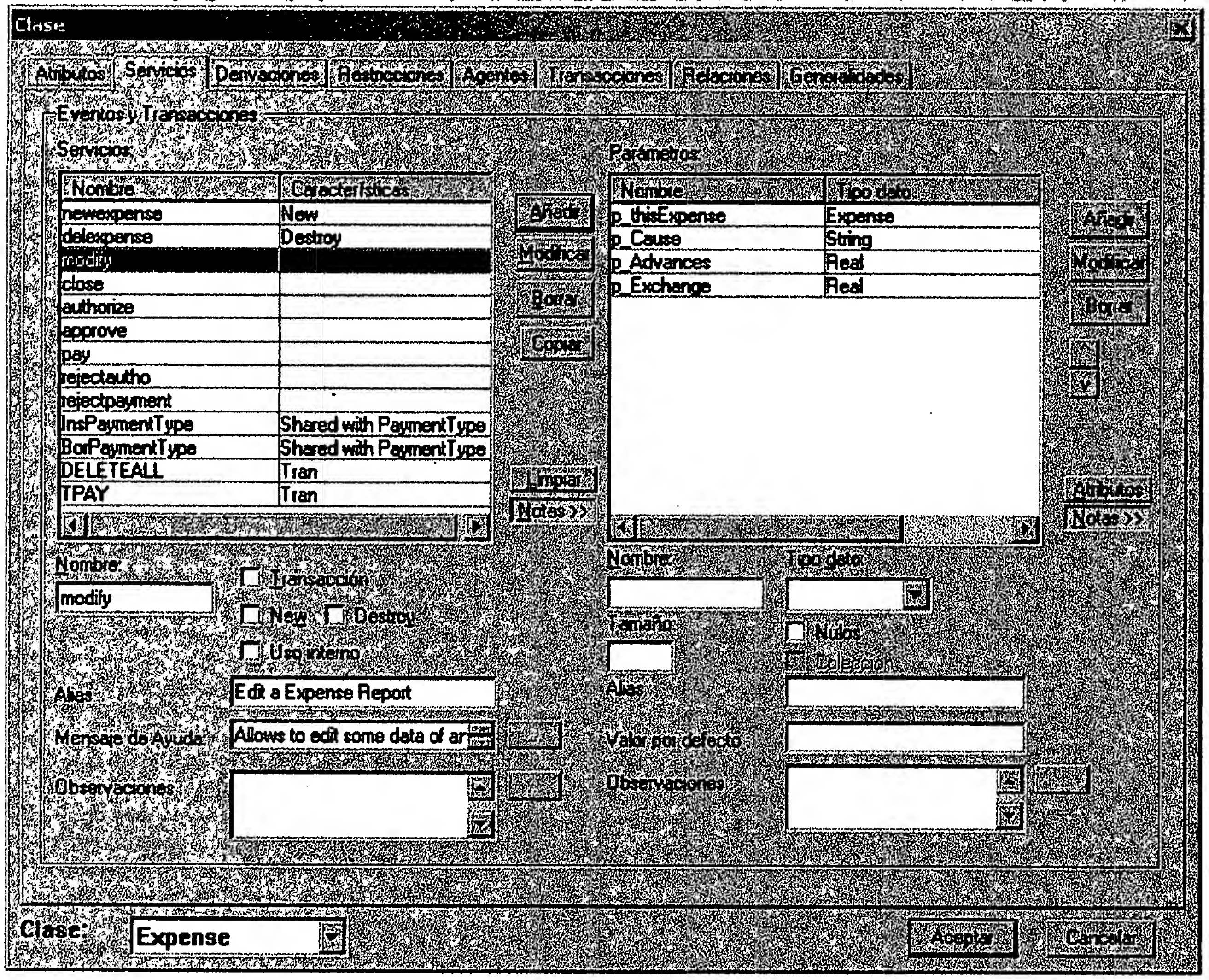
Condición:

Fórmula:

Clase: Expense

APPENDIX A

Phase 3. Filling classes' details.



Eventos y Transacciones

Servicios:

Nombre	Características
newexpense	New
delexpense	Destroy
modify	
close	
authorize	
approve	
pay	
rejectautho	
rejectpayment	
InsPaymentType	Shared with PaymentType
BorPaymentType	Shared with PaymentType
DELETEALL	Tran
TPAY	Tran

Nombre: modify

☐ Transacción
☐ New ☐ Destroy
☐ Uso interno

Alias: Edit a Expense Report

Mensaje de Ayuda: Allows to edit some data of ar

Observaciones:

Parámetros:

Nombre	Tipo dato
p_thisExpense	Expense
p_Cause	String
p_Advances	Real
p_Exchange	Real

Nombre:

Tipo dato:

Tamaño:

☐ Nulos
☐ Colección

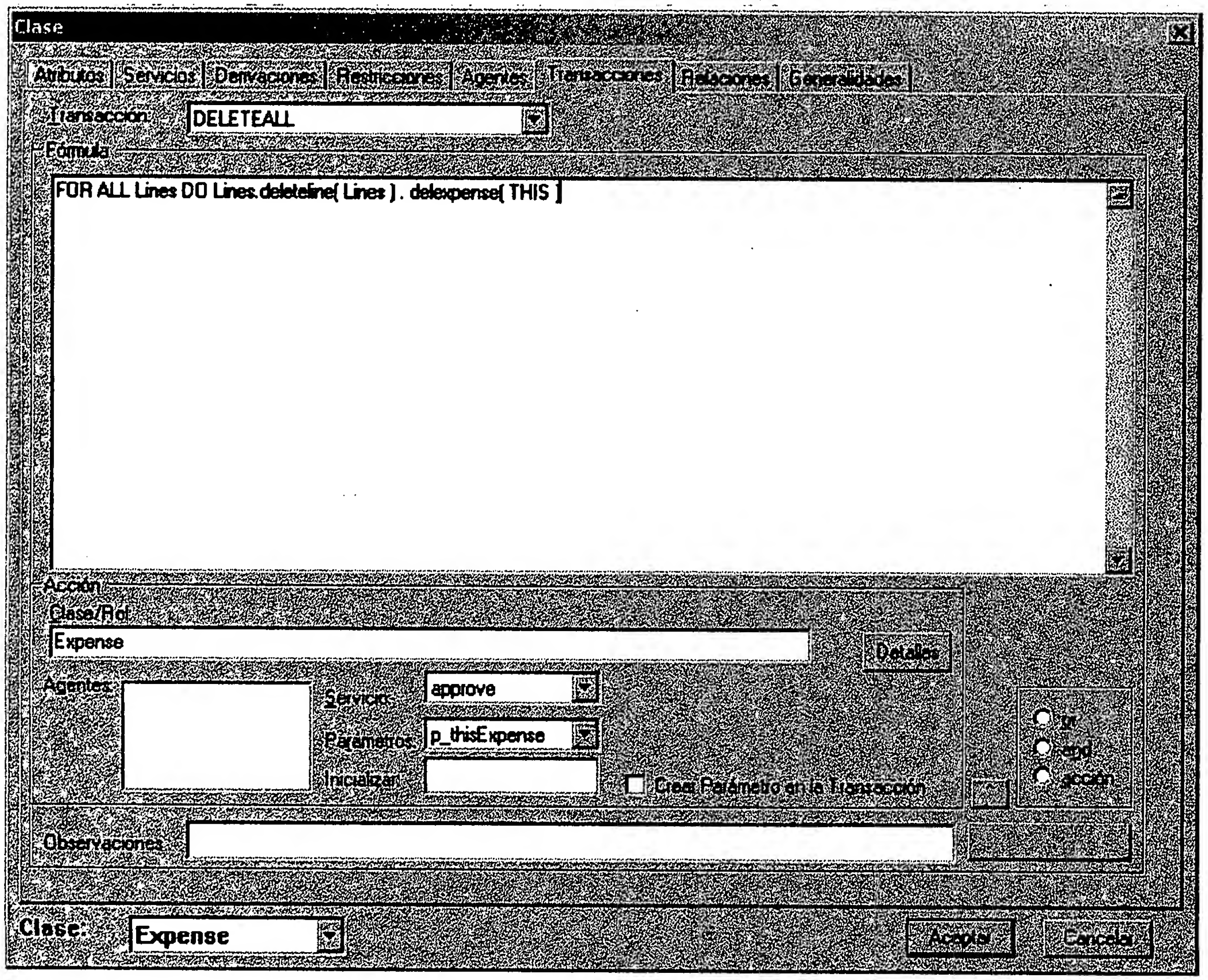
Alias:

Valor por defecto:

Observaciones:

Clase: Expense

Phase 3. Filling classes' details.



Transacción: DELETEALL

Fórmula:

```
FOR ALL Lines DO Lines.deleteline( Lines ) . delexpense( THIS )
```

Acción

Clase/Rol: Expense

Agentes:

Servicio: approve

Parámetros: p_thisExpense

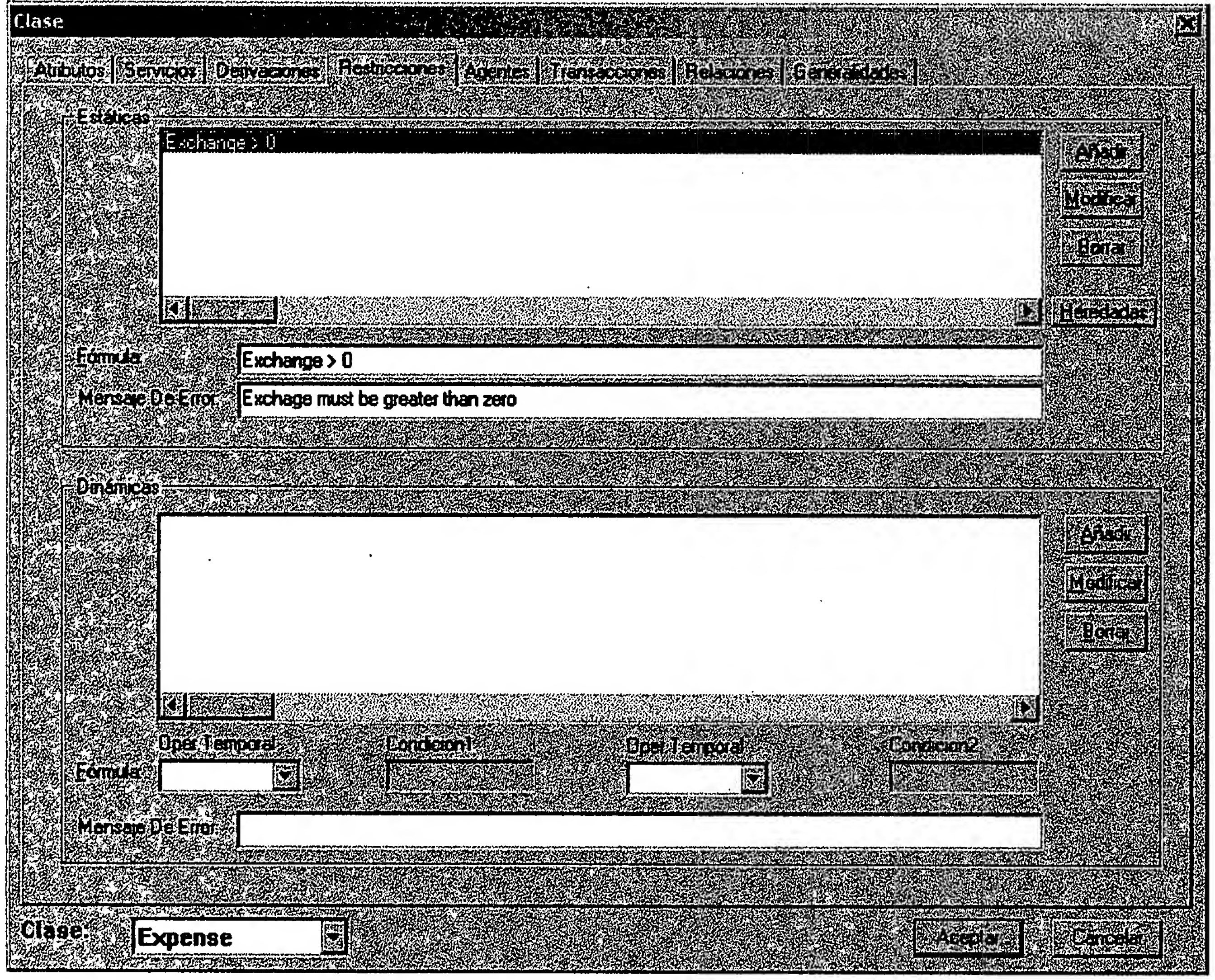
Inicializar:

☐ Crear Parámetro en la Transacción

Observaciones:

Clase: Expense

Phase 3. Filling classes' details.



Exchange > 0

Fórmula: Exchange > 0

Mensaje De Error: Exchage must be greater than zero

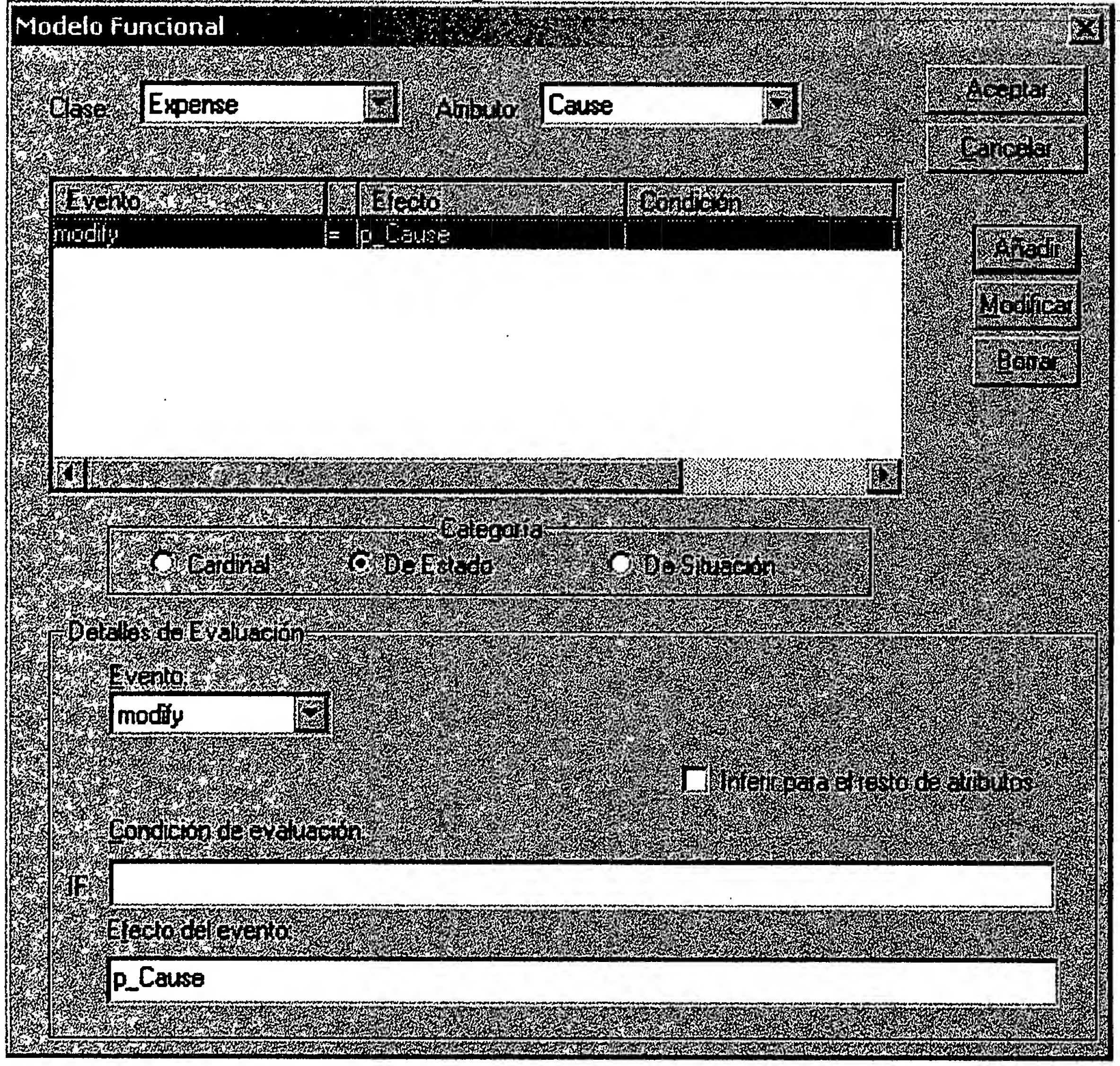
Fórmula:

Mensaje De Error:

Clase: Expense

APPENDIX A

Phase 4. Express evaluations.



Clase: Expense
Atributo: Cause

Evento		Efecto	Condición
modify	=	p_Cause	

Categoría: ( ) Cardinal [✓ De Estado] ( ) De Situación

Detalles de Evaluación

Evento: modify

☐ Inferir para el resto de atributos

Condición de evaluación:

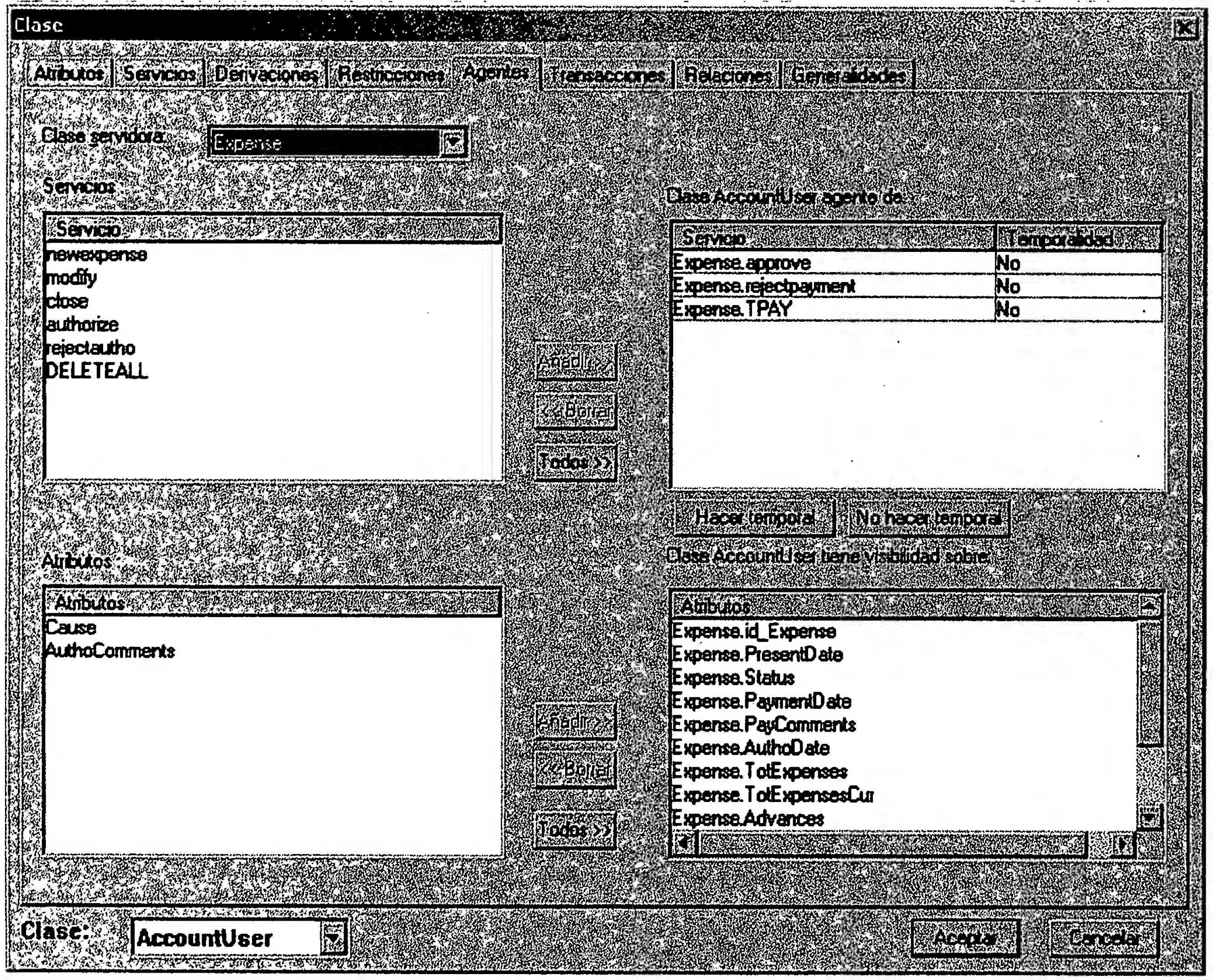
IF:

Efecto del evento:

p_Cause

APPENDIX A

Phase 5. Agent relationships.



Clase servidora: Expense

Servicios

Servicio
newexpense
modify
close
authorize
rejectautho
DELETEALL

Clase AccountUser agente de:

Servicio	Temporalidad
Expense.approve	No
Expense.rejectpayment	No
Expense.TPAY	No

Atributos

Atributos
Cause
AuthoComments

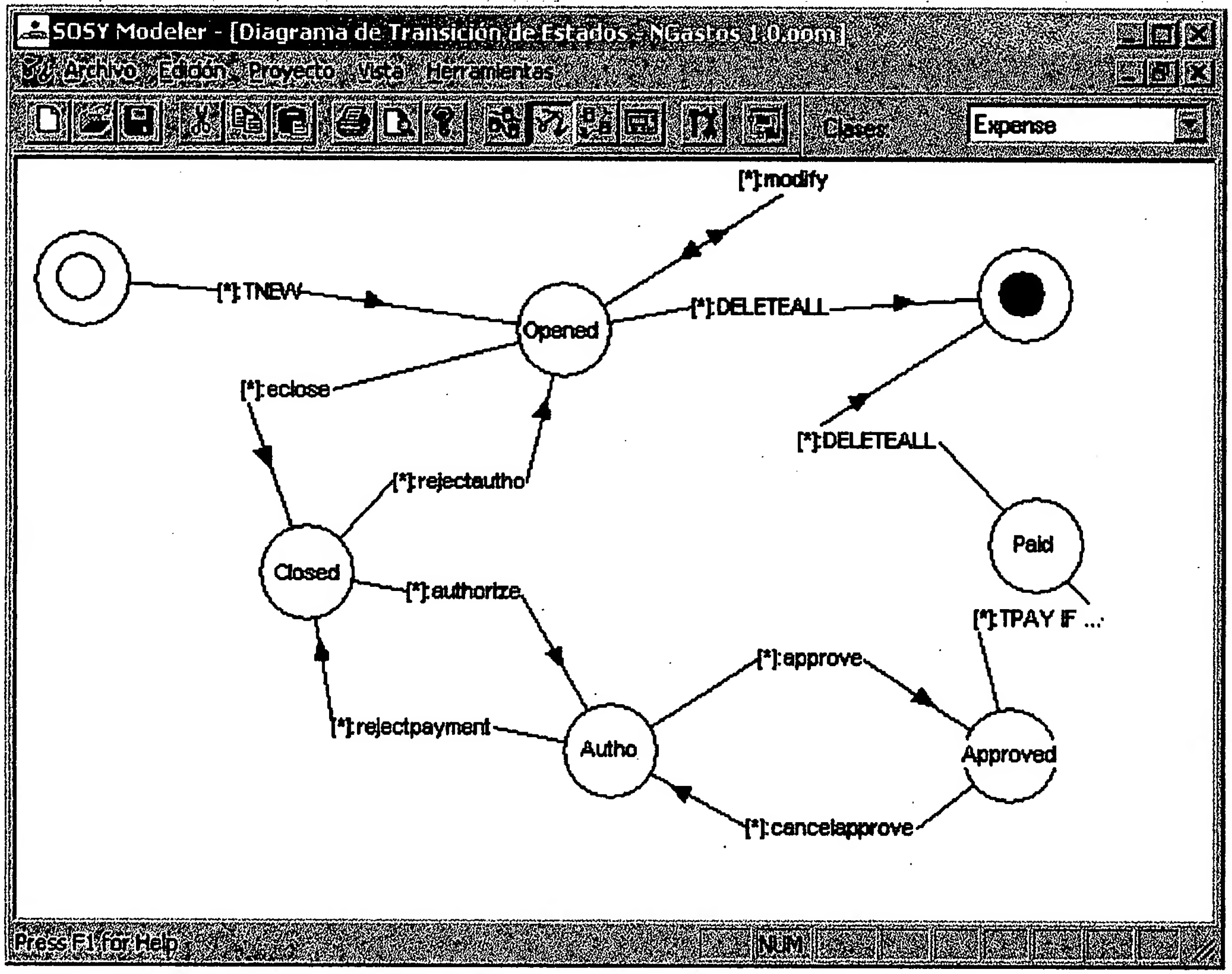
Clase AccountUser tiene visibilidad sobre:

Atributos
Expense.id_Expense
Expense.PresentDate
Expense.Status
Expense.PaymentDate
Expense.PayComments
Expense.AuthoDate
Expense.TotExpenses
Expense.TotExpensesCur
Expense.Advances

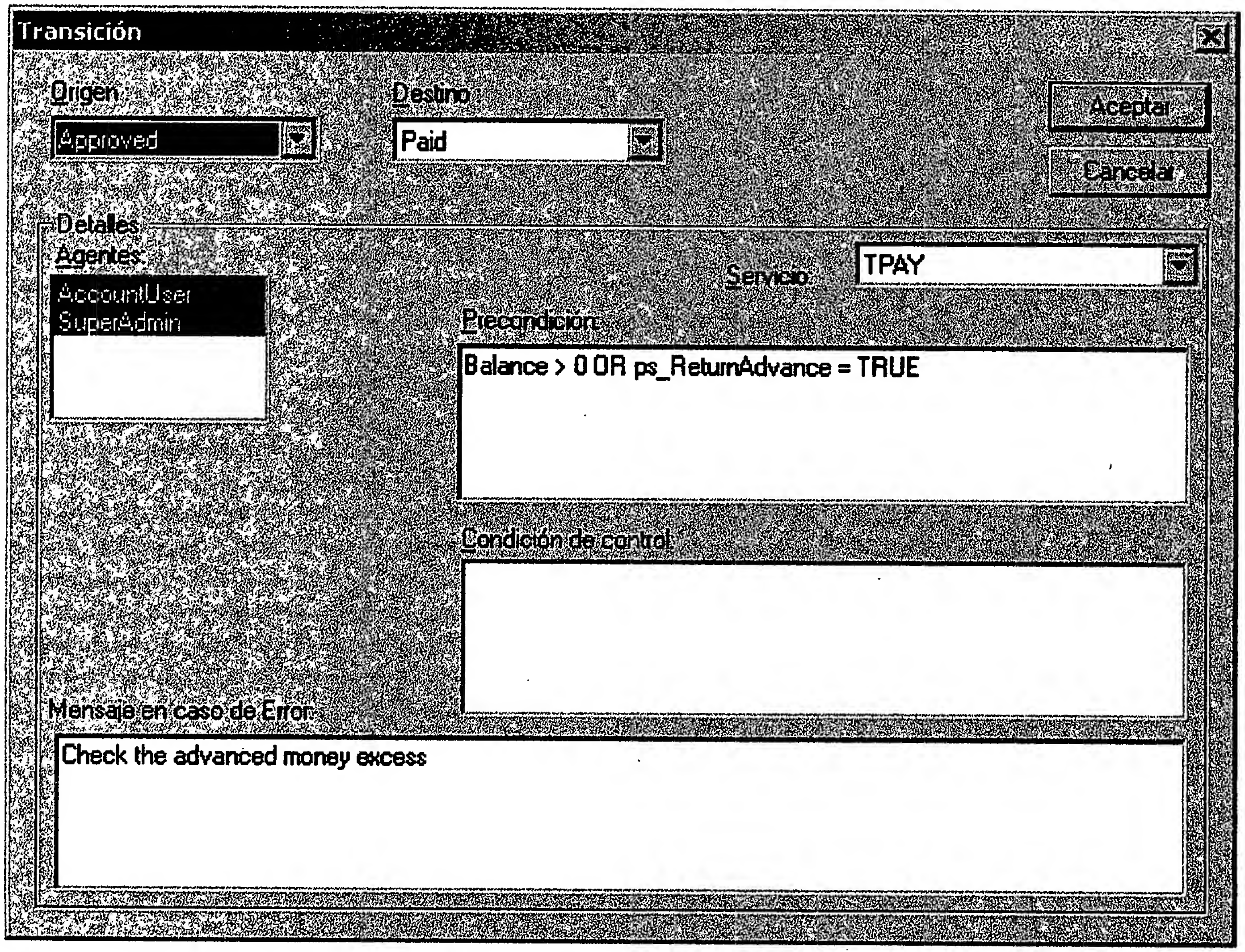
Clase: AccountUser

TOTO90" ETh2Z860

Phase 6. State Transition Diagram.



Phase 6. STD Preconditions



Origen: Approved

Destino: Paid

Agentes: AccountUser, SuperAdmin

Servicio: TPAY

Precondicion:

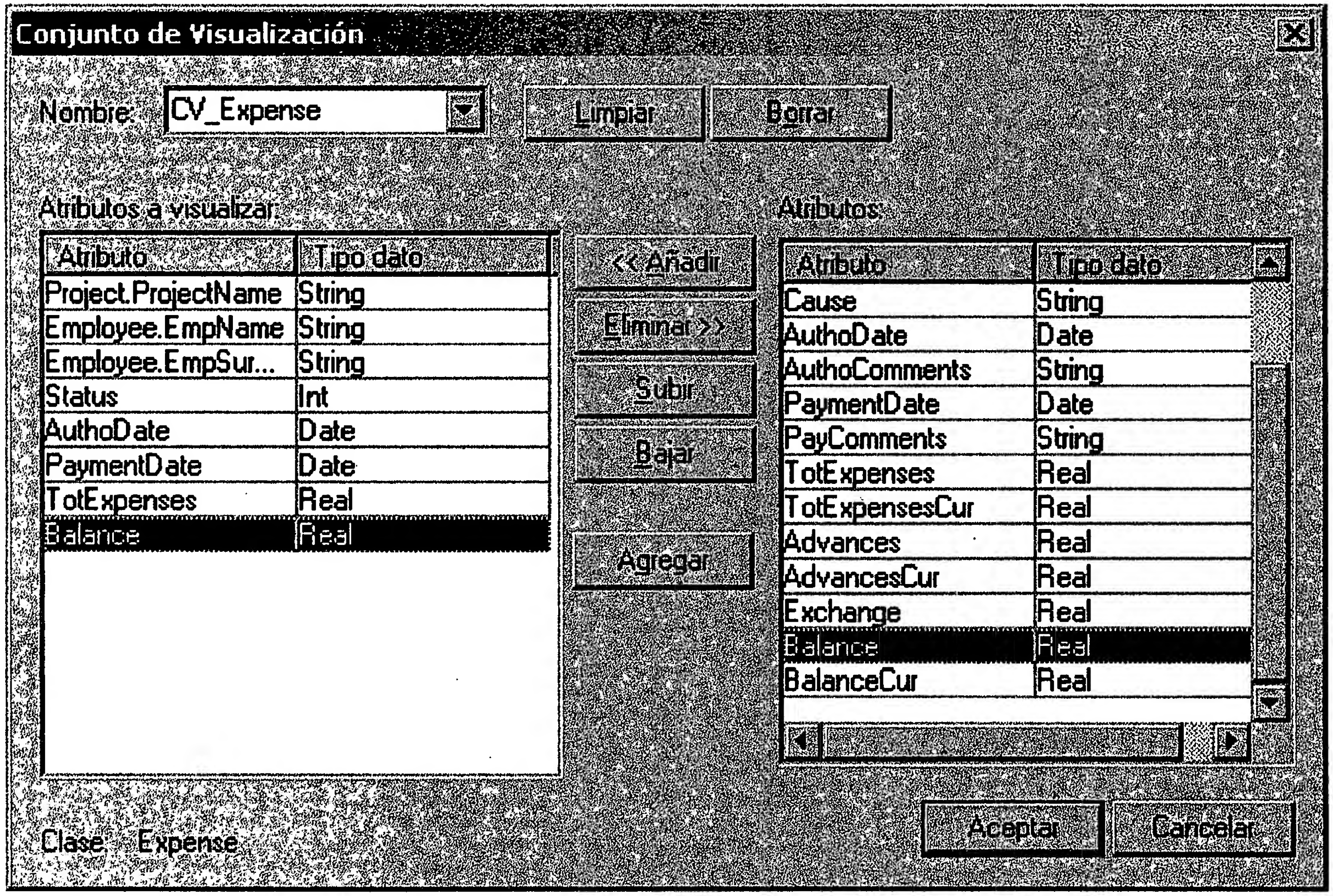
```
Balance > 0 OR ps_ReturnAdvance = TRUE
```

Condición de control:

Mensaje en caso de Error:

Check the advanced money excess

Phase 7. Presentation Model.



Conjunto de Visualización

Nombre: CV_Expense

Atributos a visualizar:

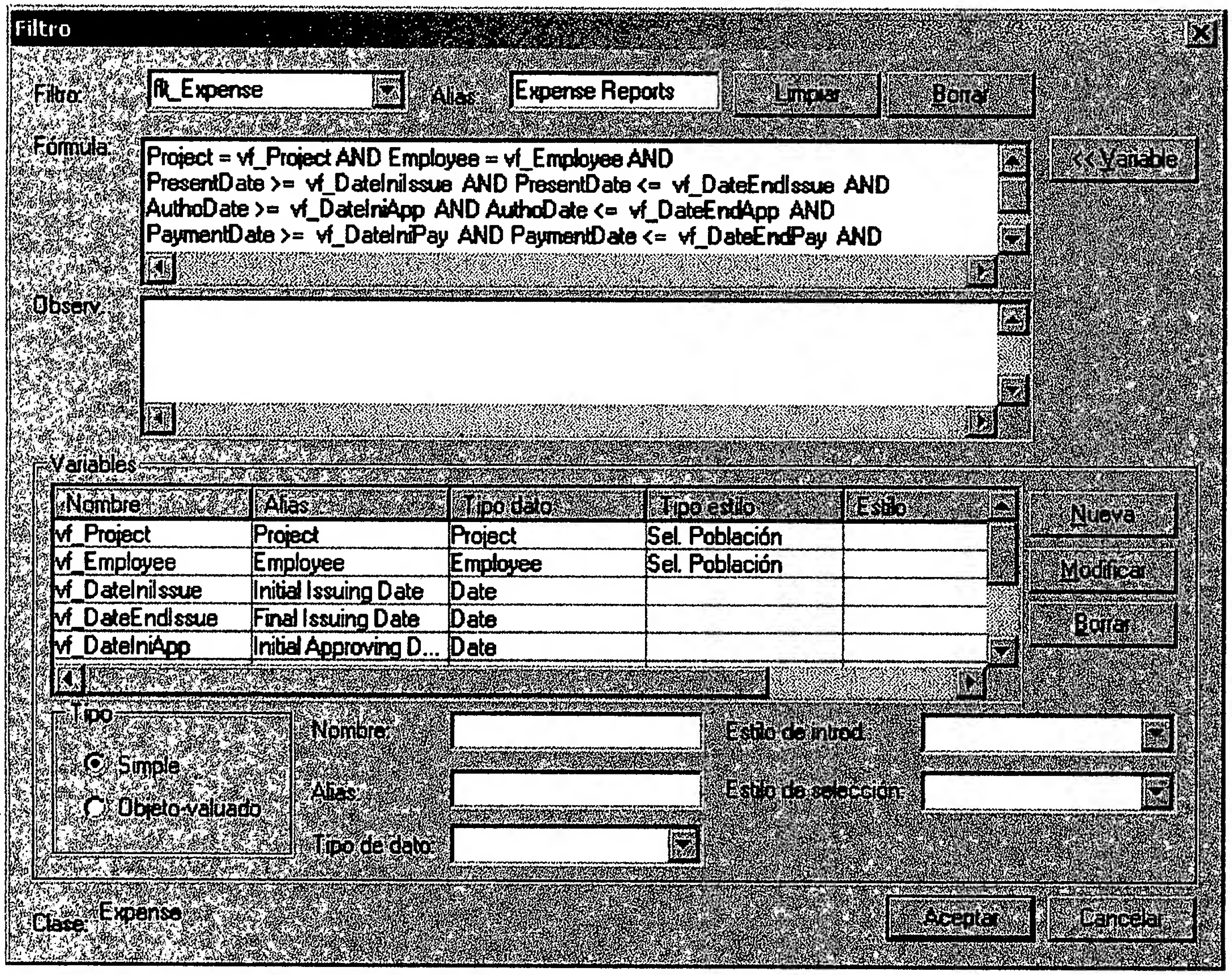
Atributo	Tipo dato
Project.ProjectName	String
Employee.EmpName	String
Employee.EmpSur...	String
Status	Int
AuthoDate	Date
PaymentDate	Date
TotExpenses	Real
Balance	Real

Atributos:

Atributo	Tipo dato
Cause	String
AuthoDate	Date
AuthoComments	String
PaymentDate	Date
PayComments	String
TotExpenses	Real
TotExpensesCur	Real
Advances	Real
AdvancesCur	Real
Exchange	Real
Balance	Real
BalanceCur	Real

Clase: Expense

Phase 7. Presentation Model.



Filtro: flt_Expense Alias: Expense Reports

Fórmula:

```
Project = vf_Project AND Employee = vf_Employee AND
PresentDate >= vf_DateIniIssue AND PresentDate <= vf_DateEndIssue AND
AuthoDate >= vf_DateIniApp AND AuthoDate <= vf_DateEndApp AND
PaymentDate >= vf_DateIniPay AND PaymentDate <= vf_DateEndPay AND
```

Observ.

Variables

Nombre	Alias	Tipo dato	Tipo estilo	Estilo
vf_Project	Project	Project	Sel. Población	
vf_Employee	Employee	Employee	Sel. Población	
vf_DateIniIssue	Initial Issuing Date	Date		
vf_DateEndIssue	Final Issuing Date	Date		
vf_DateIniApp	Initial Approving D...	Date		

Tipo: [✓ Simple] Objeto valuado

Nombre:

Alias:

Tipo de dato:

Estilo de introd.

Estilo de selección

Clase: Expense

APPENDIX A

Conceptual Model Validation

CARE Technologies, S.A.

Index

- Intro
- Overview
- Validation Degrees
  - Partial Validation
  - Total Validation

APPENDIX A

Index

- Validation Types
  - Elements of the Conceptual Model
  - Formulas of the Conceptual Model (Syntax)
- Validation Trees
  - Nodes
  - Leaves
- Example

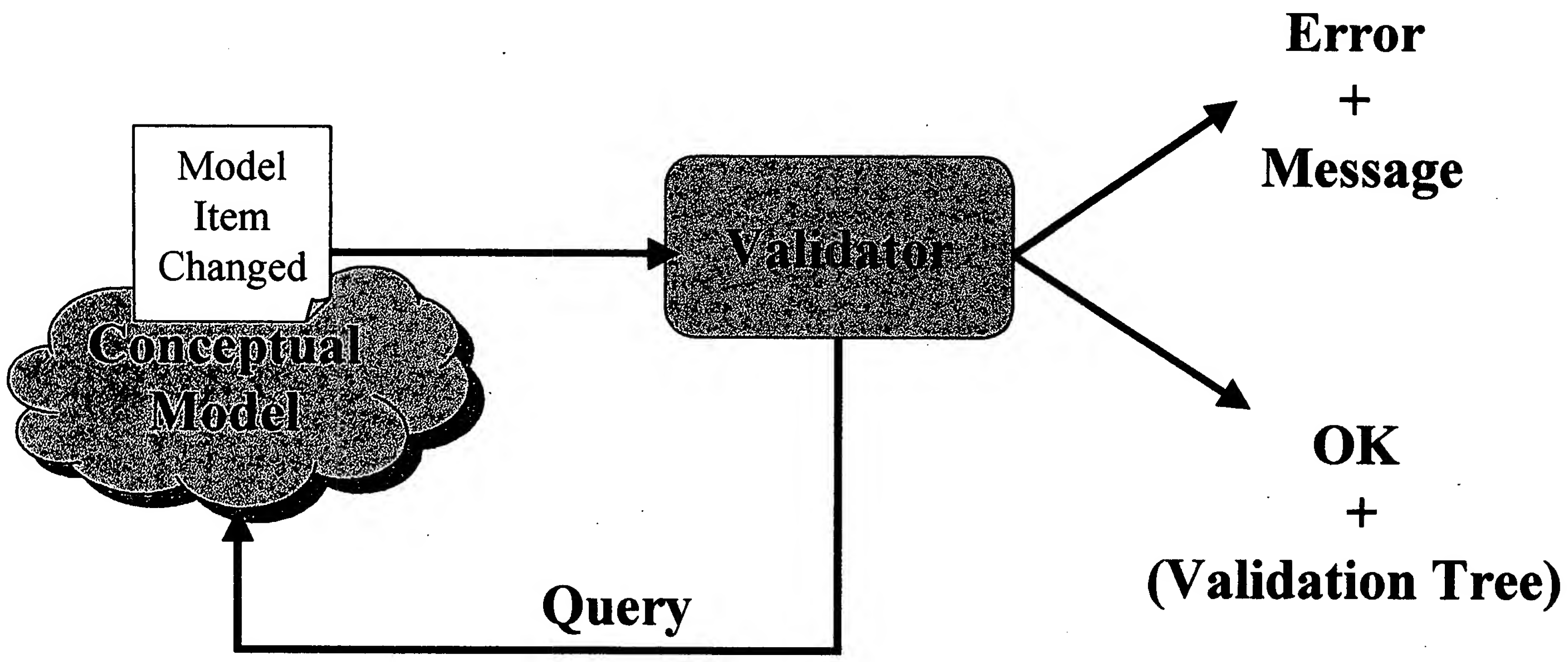
Intro

- Conceptual Model Validation is the process by which a conceptual model or a modification of it is proven to be valid:
  - Correct
    - Non Ambiguous
    - Non Contradictory
  - Complete
    - Every concept is fully specified
- Validation process checks the representation of requirements in Formal Specification Language to be valid

Validation Degrees

- Partial Validation
  - That of a single element of the Conceptual Model.
  - Happens whenever an element is added, modified or deleted.

Partial Validation Overview

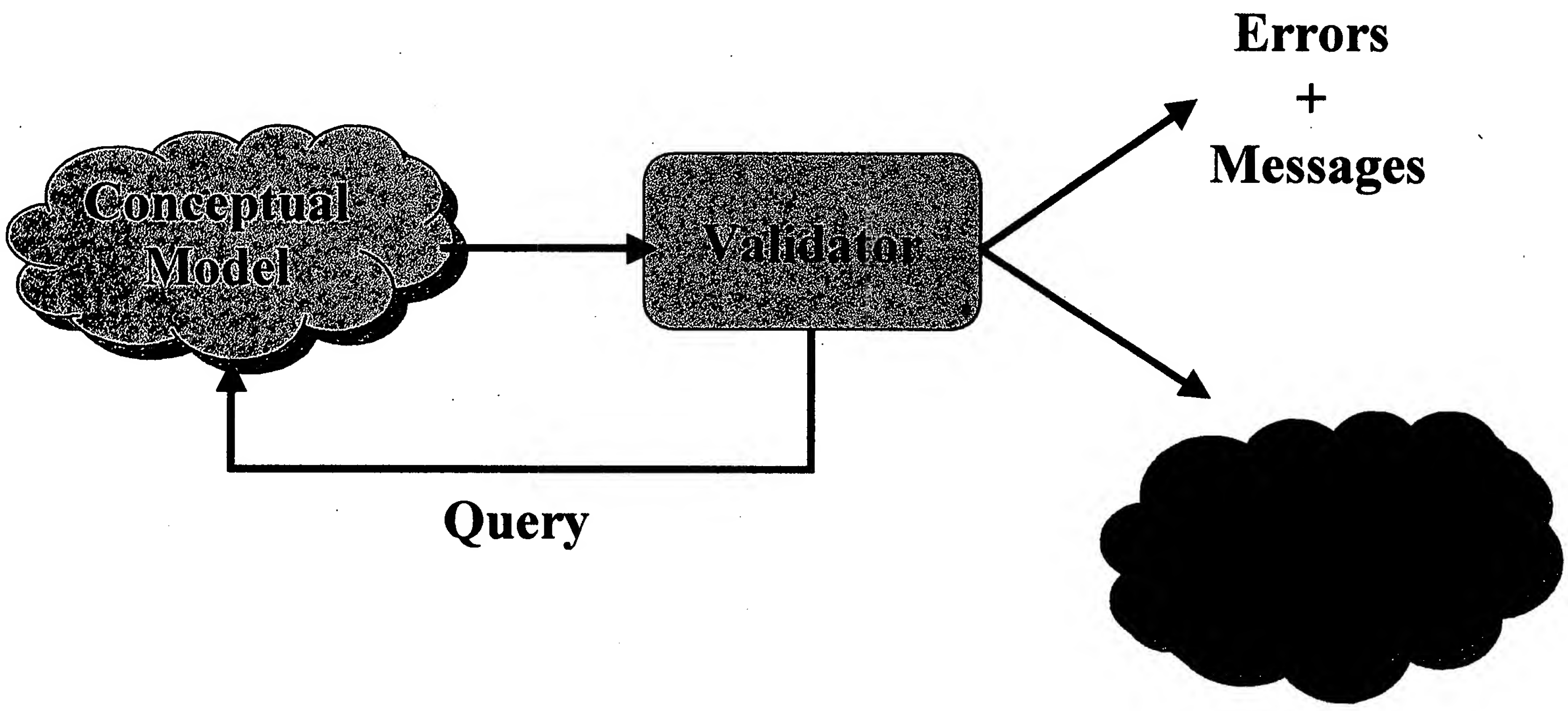


Validation Degrees

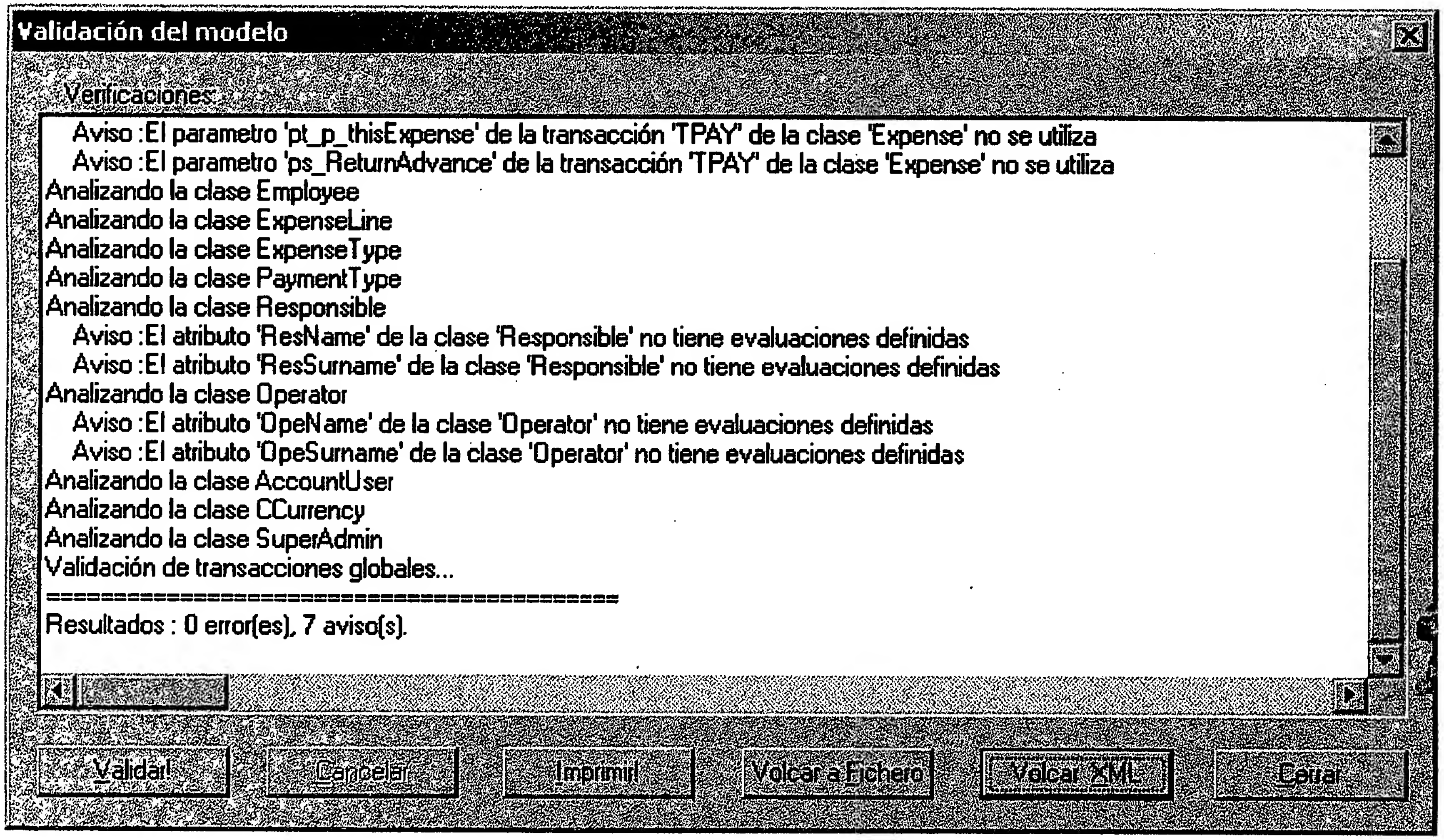
- Total Validation
  - That of the whole Conceptual Model.
  - Happens by request.
  - Must happen prior to any translation process.
  - Takes advantage of partial validations already performed.

APPENDIX A

Total Validation Overview



Total Validation Example



```
  Aviso :El parametro 'pt_p_thisExpense' de la transacción 'TPAY' de la clase 'Expense' no se utiliza
  Aviso :El parametro 'ps_ReturnAdvance' de la transacción 'TPAY' de la clase 'Expense' no se utiliza
Analizando la clase Employee
Analizando la clase ExpenseLine
Analizando la clase ExpenseType
Analizando la clase PaymentType
Analizando la clase Responsible
  Aviso :El atributo 'ResName' de la clase 'Responsible' no tiene evaluaciones definidas
  Aviso :El atributo 'ResSurname' de la clase 'Responsible' no tiene evaluaciones definidas
Analizando la clase Operator
  Aviso :El atributo 'OpeName' de la clase 'Operator' no tiene evaluaciones definidas
  Aviso :El atributo 'OpeSurname' de la clase 'Operator' no tiene evaluaciones definidas
Analizando la clase AccountUser
Analizando la clase CCurrency
Analizando la clase SuperAdmin
Validación de transacciones globales...
========================================
Resultados : 0 error(es), 7 aviso(s).
```

APPENDIX A

Validation Types

- Elements of the Conceptual Model
  - Ensure the properties of an element (except formulas) are correct and complete.
  - Conditions that must hold depend on the type of element and the property being validated.
  - Examples:
    - Class Name is unique in a Conceptual Model.
    - Attribute Name is unique in its Class (but not in a Conceptual Model)

Validation Types

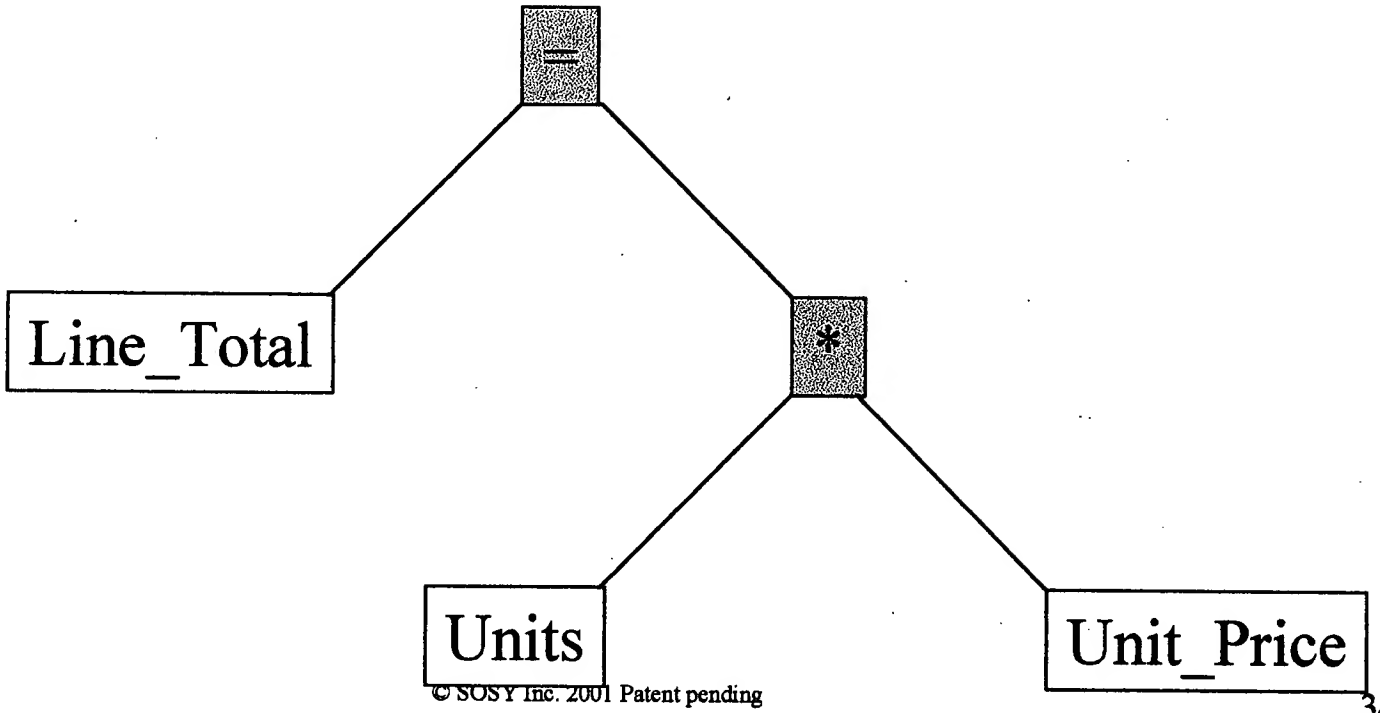
- Formulas of the Conceptual Model
  - Ensure the formulas of the Conceptual Model are correct and complete.
  - Syntactical and Semantical Validation according to an extended Formal Specification Language grammar.
  - Input:
    - Formula expression
    - Formula Type (precondition, valuation, …etc.)
    - Formula Context (class name, service name, …etc.)
  - Output:
    - Error Message (validation did not pass)
    - Validation Tree (validation passed)

Validation Trees

- Binary Tree representation of a correct formula.
- Tree consists of Nodes and Leaves.
- Nodes
  - Represent operators
  - Can have one or two “branches” (binary)
  - Branches can again be nodes or leaves
- Leaves
  - Represent operands
  - Have no branches

Example

- Line_Total = Units * Unit_Price



APPENDIX A

Documentation Translation

CARE Technologies, S.A.

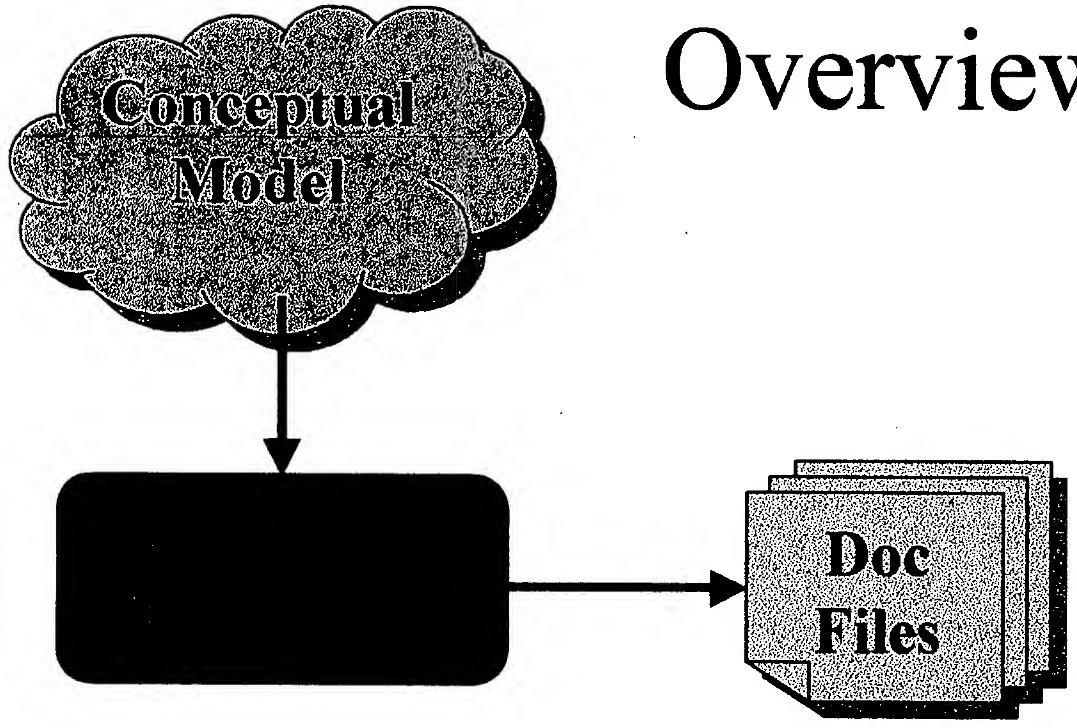
Index

- Intro
- Overview
- Output Detail
  - Document Types
  - Document Formats
- Translation
  - CM Subset of Interest
  - Translation Process
  - Remarks
- Example

Intro

- Documentation Translation is the process to obtain, from a Conceptual Model, documentation on the system it represents.
- Documentation can have several degrees of detail and be focused on different aspects, thus obtaining different documentation formats from the same Conceptual Model.

Overview



APPENDIX A

Document Type

- Help
- Full
- General
- User Help Manual
- Project Report
- Test Report

Document Format

- Multifile HTML
- Single File HTML
- ASCII Text
- LaTeX
- RTF
- Compiled HTML

Output Detail

- Document Types
  - Help
    - Description of each Class, its Attributes, Services and Population Selection Filters.
  - Full
    - Full description of a Conceptual Model
    - Aimed at analysts.
  - General
    - Description of each Class Attributes, Identification Function, Services, Aggregation Relationships and Specialization Relationships.

Output Detail

- Document Types
  - User Help Manual
    - Both Help Manual and Contextual Help (F1 key).
    - Intended for Operation Manual.
    - Integration with User Interface applications.
  - Project Report
    - Description of each Class Attributes and Services.
  - Test Report
    - Description of each Class Services.
    - Intended for Testing purposes.

Output Detail

- Document Formats
  - Multifile HTML
    - One HTML page per concept.
    - Recommended for navigable help.
  - Single File HTML
    - One single HTML page.
    - Recommended for printing.
  - ASCII Text
    - Single, plain ASCII text file.

Output Detail

- Document Formats
  - LaTeX
    - Single, LaTeX text file.
  - RTF
    - Single, RTF text file.
  - Compiled HTML
    - Same as Multifile HTML plus header files to be used by HTML Help Workshop compiler.
    - Recommended for contextual help.
    - Searching and Indexing facilities usage from browsers.

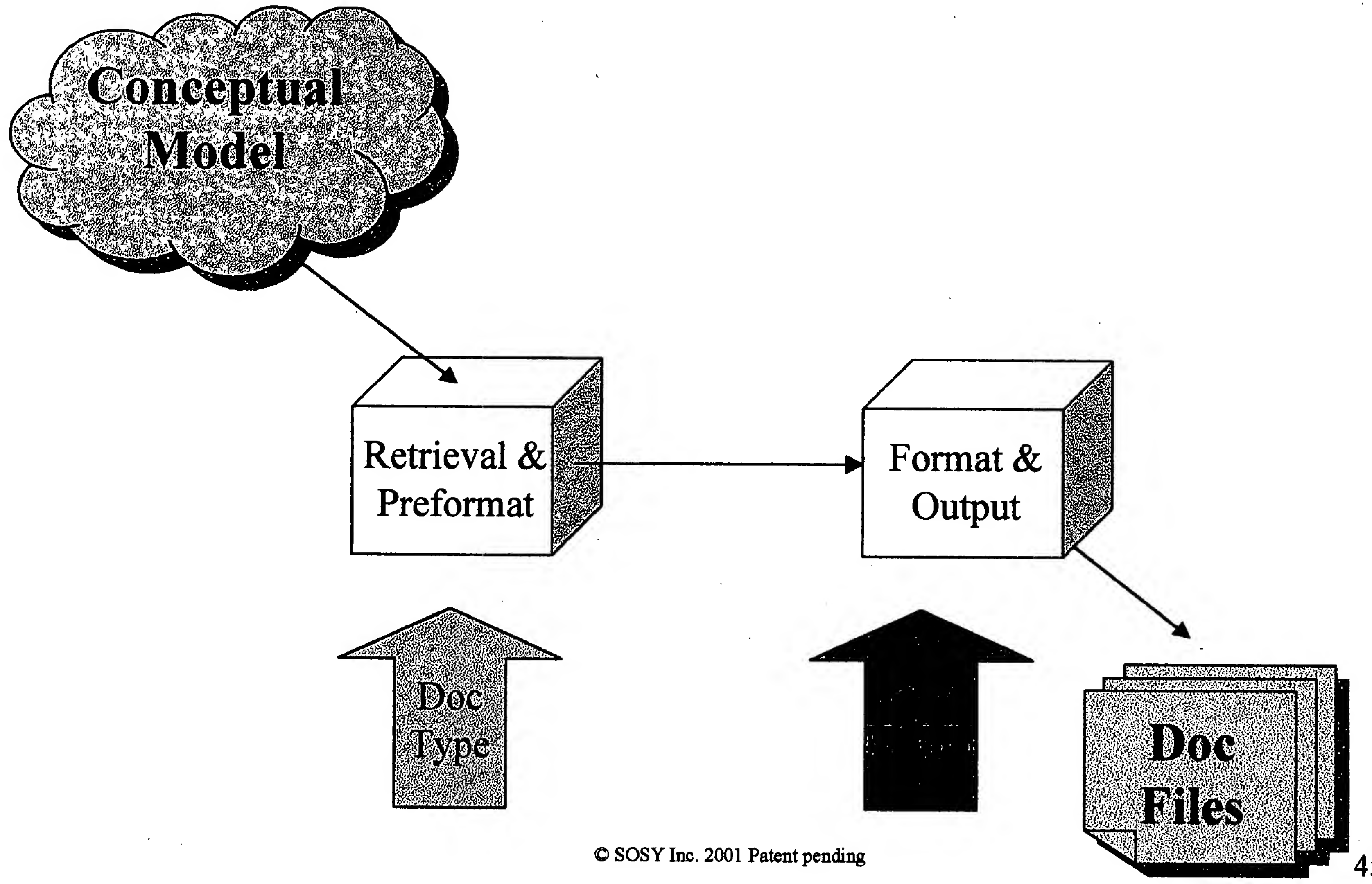
Translation

- Conceptual Model Subset of Interest
  - Subset of Interest depends on Document Type.
  - Usual elements:
    - Classes
    - Attributes
    - Relationships
    - Services & Arguments
  - Intensive use of analysis information.

Translation

- Translation Process
  - Read information from Conceptual Model and format it for output.
  - Two phases:
    - Information retrieval and pre-formatting.
      - Depends on Document Type
      - Independent from Document Format
    - Information output.
      - Depends on Document Format.
      - Independent from Document Type.

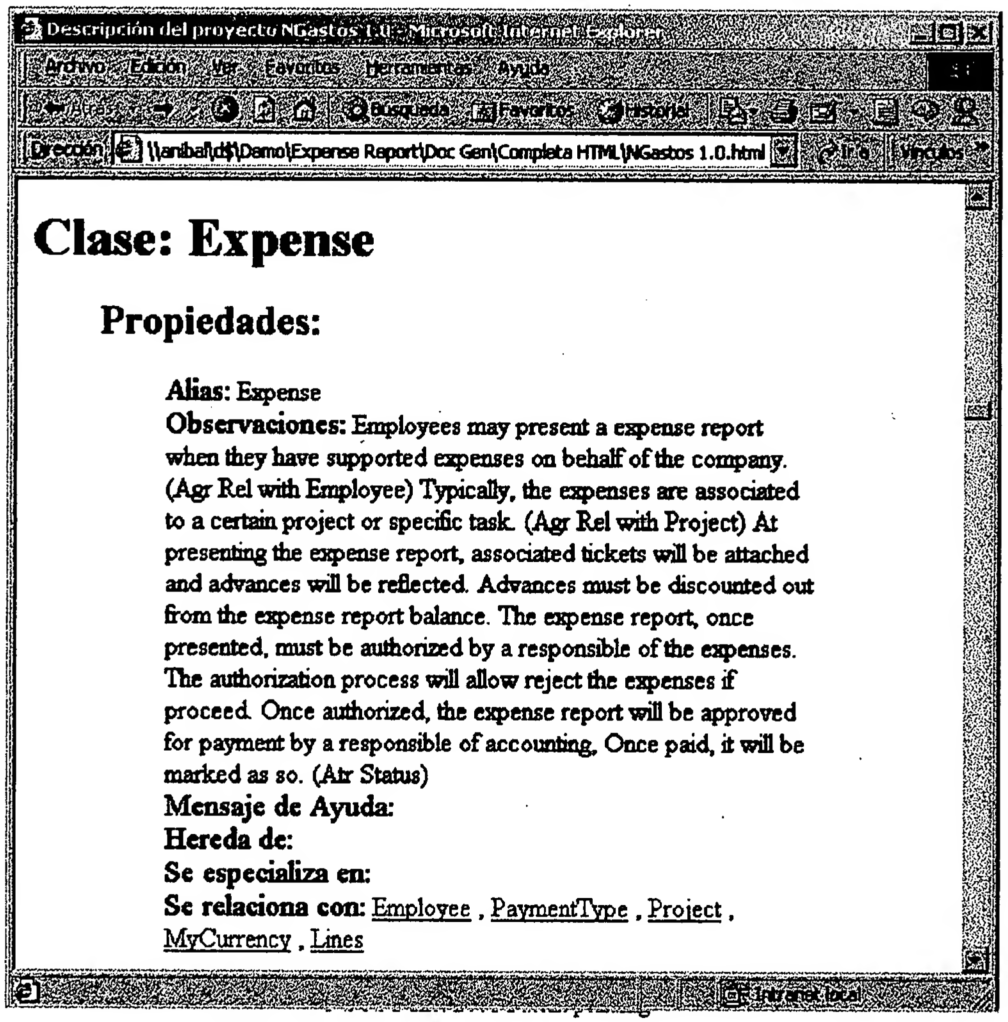
Translation Phases



Translation

- Remarks
  - Conceptual Model needs not to be valid (in terms of completeness and correctness) but it is always non-ambiguous.
  - The richer the analysis information, the richer the documentation.
  - Easily extensible
    - New Document Types
    - New Document Formats

Example


Descripción del proyecto NGastos 1.0 - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Búsqueda Favoritos Historial
Dirección \\aniba\d$\Damo\Expense Report\Doc Gen\Completa HTML\NGastos 1.0.html
Ir a Vínculos
Intranet local

Clase: Expense

Propiedades:

Alias: Expense
Observaciones: Employees may present a expense report when they have supported expenses on behalf of the company. (Agr Rel with Employee) Typically, the expenses are associated to a certain project or specific task. (Agr Rel with Project) At presenting the expense report, associated tickets will be attached and advances will be reflected. Advances must be discounted out from the expense report balance. The expense report, once presented, must be authorized by a responsible of the expenses. The authorization process will allow reject the expenses if proceed. Once authorized, the expense report will be approved for payment by a responsible of accounting. Once paid, it will be marked as so. (Atr Status)
Mensaje de Ayuda:
Hereda de:
Se especializa en:
Se relaciona con: Employee , PaymentType , Project , MyCurrency , Lines

APPENDIX A

APPENDIX A

Persistence Relational Database Translation

CARE Technologies, S.A.

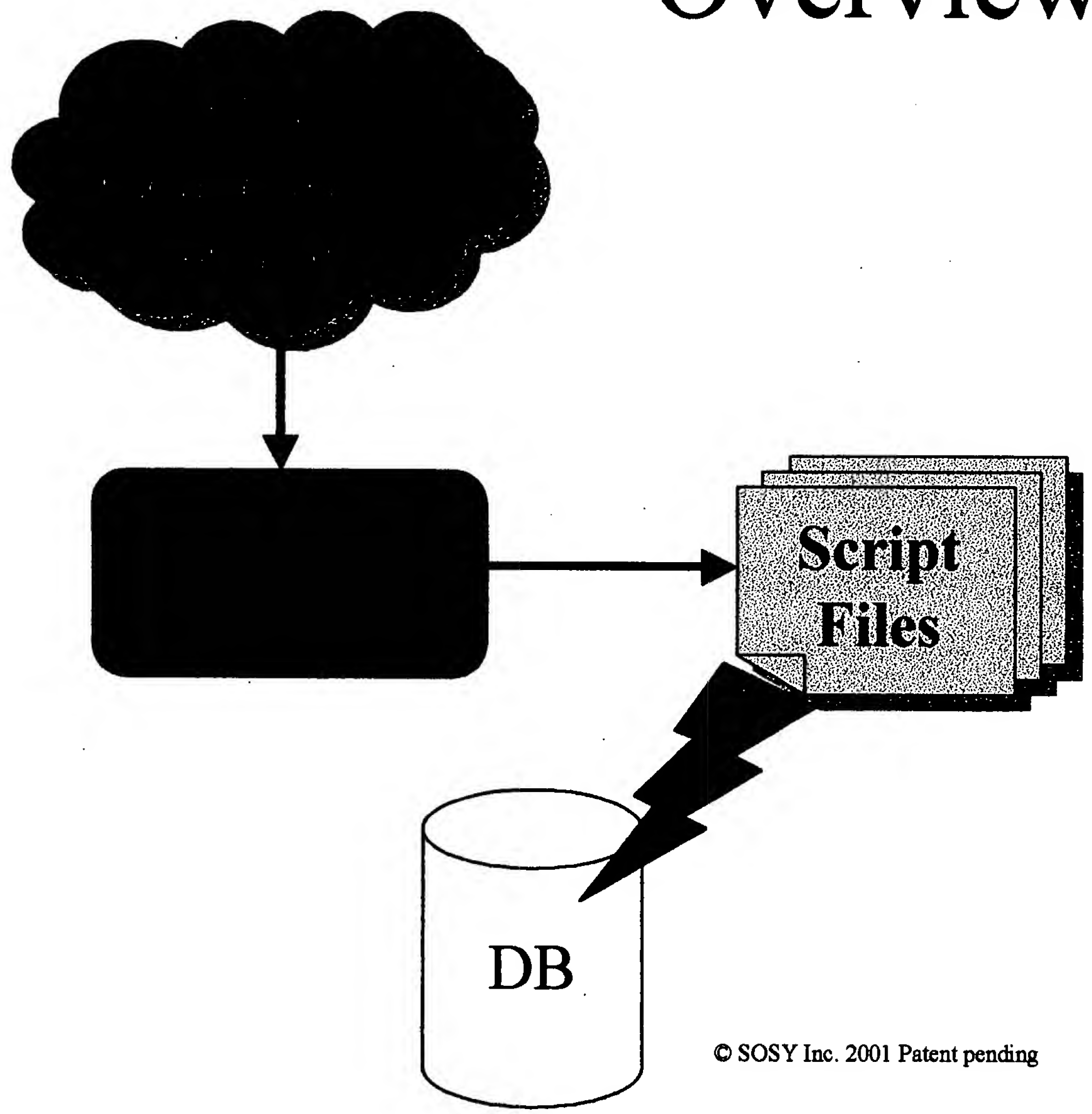
Index

- Intro
- Overview
- Output Detail
- Translation
  - – CM Subset of Interest
  - – Translation Processes
- Example

Intro

- Persistence Relational Database Translation is the process of creating a Relational Database from a certain subset of information in the Object Model of a valid Conceptual Model.
- Output script files are used to create a relational database using structured query language (SQL).

Overview



- Creates
- Primary Keys
- Foreign Keys
- Indexes
- Drop Creates
- Drop Primary Keys
- Drop Foreign Keys
- Drop Indexes

Output Detail

- Creates
  - Creation of Tables and Fields
- Primary Keys
  - Creation of Primary Keys as constraints on each table
- Foreign Keys
  - Creation of Foreign Keys as constraints on each table
- Indexes
  - Creation of Indexed on each table

Output Detail

- Drop Creates
  - Deletion of Tables
- Drop Primary Keys
  - Deletion of Primary Key Constraints
- Drop Foreign Keys
  - Deletion of Foreign Key Constraints
- Drop Indexes
  - Deletion of Indexes

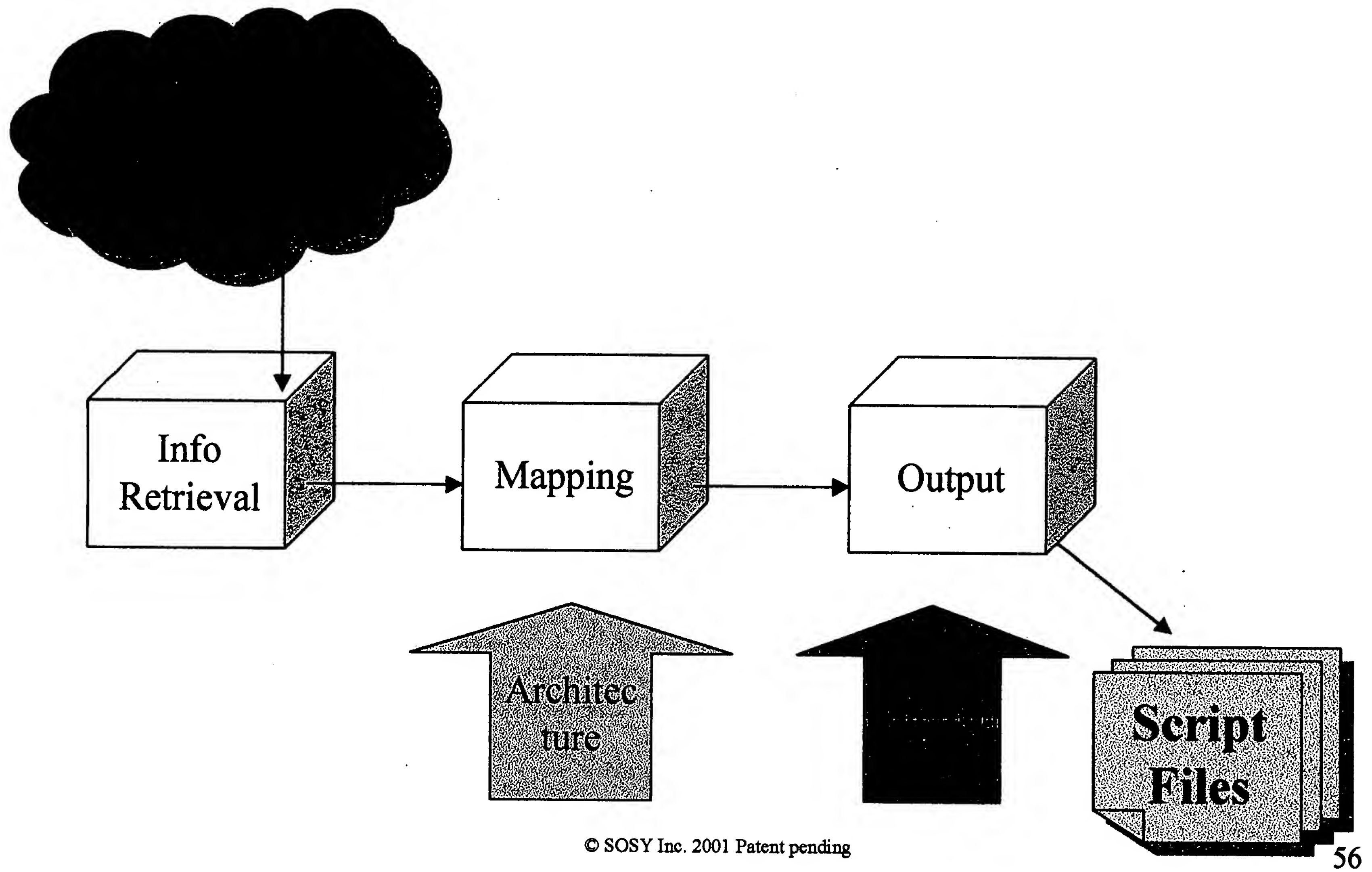
Translation

- Conceptual Model Subset of Interest
  - Object Model
    - Classes
    - Attributes
    - Identification Functions
    - Aggregation Relationships
    - Inheritance Relationships

Translation

- Three phases:
  - Information retrieval.
    - Independent from persistence architecture.
  - Fixed architecture mapping.
    - Depends on persistence architecture.
  - Information output.
    - Targeted for Standard ANSI SQL 92 RDBMS.
    - Script files depends on the platform's SQL syntax of RDBMS manufacturer.
    - May depend on platform specifications to make use of manufacturer extensions and tuning.

Translation Phases



Translation

- Translation Processes. Mapping:
  - Class → Table
  - Non-derived Attribute → Field
  - Identification Function → Primary Key
  - Univaluated Relationship →Foreign Key
  - Univaluated Relationship → Index
  - Multivaluated Relationship → Table
  - Inheritance Relationship →Foreign Key

Example

Create table script in SQL for Expense class

```
CREATE TABLE Expense (
        fk_Project_1 int NOT NULL ,
        id_Expense int NOT NULL ,
        fk_Employee_1 CHAR(10) NOT NULL ,
        fk_MyCurrency_1 CHAR(5) NOT NULL ,
        fk_PaymentType_1 CHAR(5) NULL ,
        PresentDate datetime NOT NULL ,
        Status int NOT NULL ,
        Cause VARCHAR(255) NOT NULL ,
        AuthoDate datetime NULL ,
        AuthoComments VARCHAR(255) NULL ,
        PaymentDate datetime NULL ,
        PayComments VARCHAR(255) NULL ,
        Advances DECIMAL(19,6) NOT NULL ,
        Exchange DECIMAL(19,6) NOT NULL);
```

APPENDIX A

Business Logic Translation

CARE Technologies, S.A.

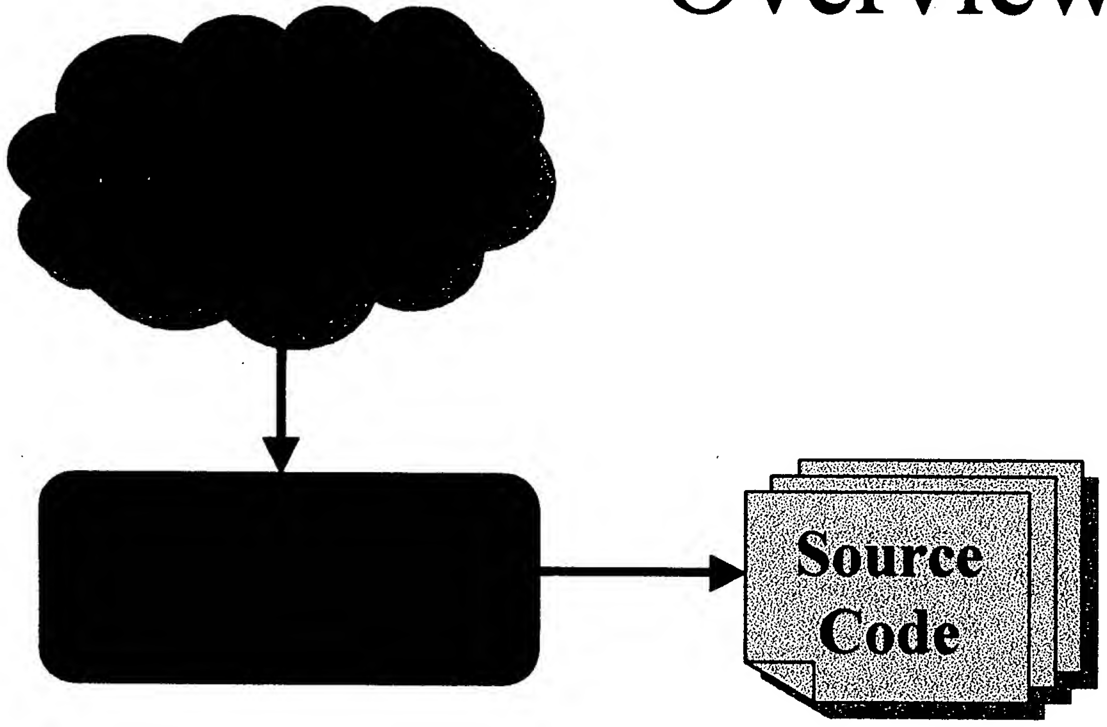
Index

- Intro
- Overview
- Output Detail
- Translation
  - CM Subset of Interest
  - Translation Processes
- Example

Intro

- Business Logic Translation is the process to obtain, following a precise Execution Model, the source code corresponding to the business logic from a valid Conceptual Model for a target Programming Language and Software Architecture.
- Execution Model is independent from Programming Language and Software Architecture.

Overview



Determines:

-Target Programming Language

-Target Software Architecture

Output Detail

- Target Programming Language and Software Architecture determine:
  - Source code organization in files
  - Files internal organization
- Source Code's backbone: Execution Model.

Output Detail

- Traceability: Source code highly readable and maintainable thanks to:
  - Source code is always organized and structured in the same way.
  - Naming conventions applied.
  - Source code includes analysis information from the Conceptual Model as comments.

Output Detail

- Implementation of a precise Execution Model grants Functional Equivalence with Conceptual Model.
- Programming Interface to Clients for:
  - Actor Validation and Authentication.
  - Services Execution.
  - Queries Execution.
- Manages:
  - Concurrency.
  - Transactions.
  - Interoperable Object Persistence.

Translation

- Conceptual Model Subset of Interest
  - Object Model
    - Static properties (Visibility & Persistence)
      - Attributes + Identification Functions
      - Derivations
      - Aggregation Relationships
      - Inheritance Relationships
    - Services (Execution Model)
      - Arguments
      - Preconditions
      - Transaction Formulas
    - Actors (Execution Model)
    - Integrity Constraints (Execution Model)

Translation

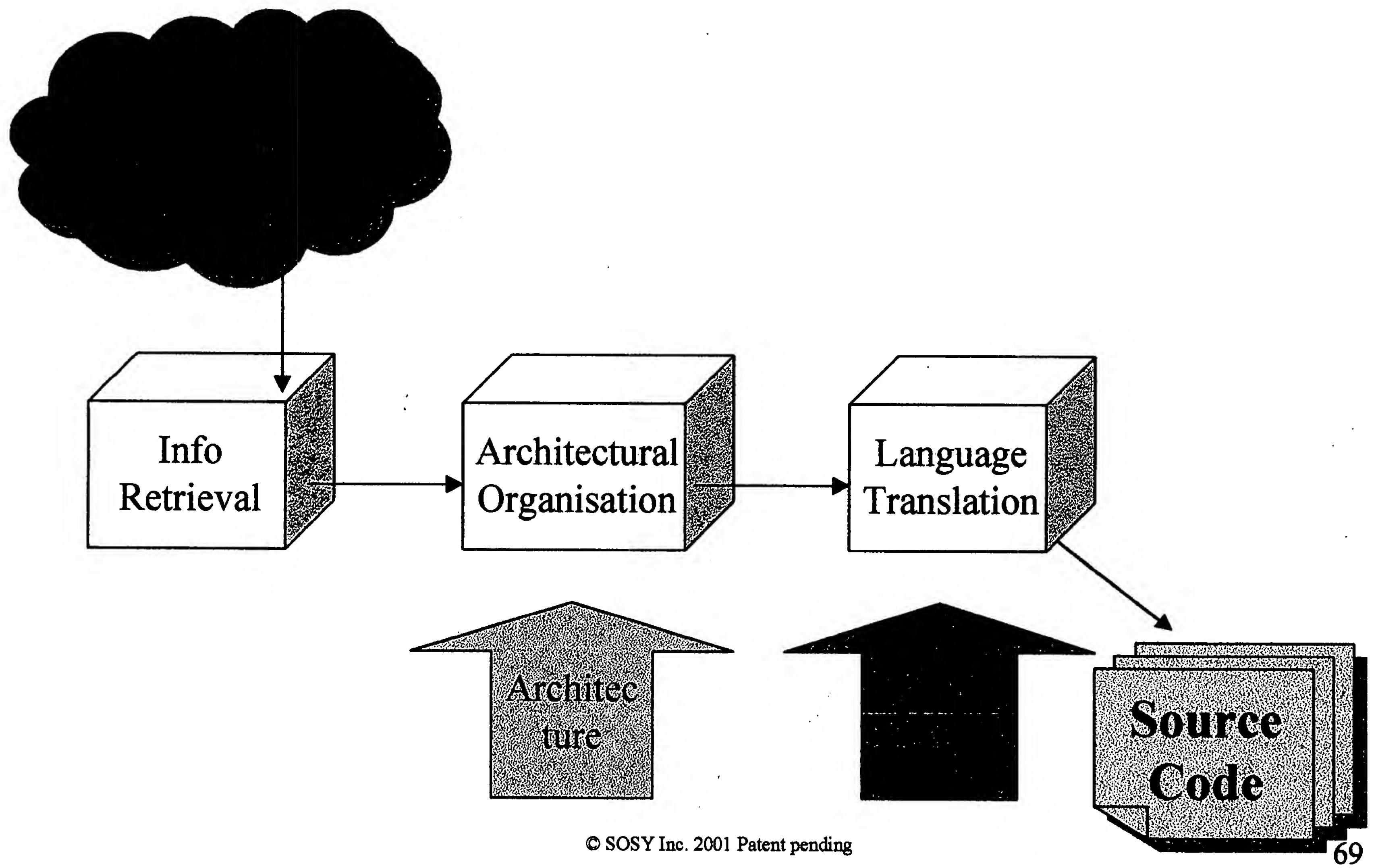
- Conceptual Model Subset of Interest.
  - Dynamic Model.
    - State Transition Diagram (Execution Model).
      - Controls Valid Lifes for an Object.
    - Object Interaction Diagram.
      - Triggers (Execution Model).
      - Global Transactions (Execution Model).
  - Functional Model (Execution Model).
    - Object state change upon occurrence of an event.

Translation

- Translation phases:
  - Information retrieval
    - Independent from target Software Architecture and Programming Language
  - Architectural organisation
    - Depends on target Software Architecture
    - Independent from target Programming Language
    - Determines files organisation and files internal structure
  - Language translation
    - Depends on target Programming Language
    - Influenced by Software Architecture
    - Takes advantage of Programming Language capabilities

09872413 .060101

Translation Phases



APPENDIX A

Translation

- Translation Processes
  - Classes
    - Static properties translation
    - Services translation
    - Queries translation
  - Global Interactions
    - Services translation
  - Global Functions
    - Functions Interface translation
    - Body is left blank

Example

- Evaluation:
  - Service Authorize modifies attributes Status, AuthoDate and AuthoComments
    - Formal Specification Language expression for evaluation

      Valuation

      [authorize ()] Status=2 and AuthoDate=today() and AuthoComments="";

    - Visual Basic Produced

```
Private Function MV_Eval_Expense_authorize() As String
     Expense_Status = 2
     Expense_AuthoDate = today()
    Expense_AuthoComments = ""
    MV_Eval_Expense_authorize = ""
End Function
```

APPENDIX A

User Interface Translation

CARE Technologies, S.A.